Cureus Journal of Computer Science
https://doi.org/10.7759/s44389-026-00193-w

Original Article

Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition

Muhammad Munsarif [1], Richmond Ampah-Mensah [1]

1. *Informatika, Universitas Muhammadiyah Semarang, Semarang, IDN*

Received: June 07, 2026 | Review began: July 13, 2026 | Review ended: July 27, 2026 | Published: July 30, 2026

© **Copyright** 2026
This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

Handwritten digit recognition remains useful for studying compact image-classification models, although the MNIST (Modified National Institute of Standards and Technology) benchmark is no longer sufficient for claiming state-of-the-art novelty. Accordingly, this study frames MNIST as a controlled setting for analyzing accuracy-efficiency behavior in lightweight convolutional neural networks (CNNs). It examines how optimizer choice, epoch count, data augmentation, and convolutional parameter settings jointly affect classification accuracy and execution time. The experimental design covers two 5-layer CNN configurations, five parameter-varying cases, eight named optimizer settings, one baseline control setting, five epoch values, and two augmentation conditions. In total, 540 paired observations of accuracy and execution time are analyzed. The results show that peak accuracy and practical efficiency lead to different conclusions. The best observed accuracy was 99.50% for the standard CNN-5LS configuration with Nadam at 120 epochs after augmentation, but this required 1,011 s. RMSprop reached 99.47% after augmentation at both 15 and 120 epochs in the standard configuration, while the 15-epoch setting required only 114 s. In the parameter-varied CNN-5L model, Case-1 was generally the most efficient and stable configuration, while wider dilation and stride combinations frequently reduced reliability. Data augmentation improved weak or unstable settings more strongly than already stable configurations, and Ftrl did not produce meaningful learning in these experiments. These findings suggest that lightweight CNN design for handwritten digit recognition should be evaluated through accuracy-time trade-offs, not only through maximum accuracy. The scope is practical configuration guidance within the MNIST experimental setting.

Categories: Computer Vision, AI applications, Machine Learning (ML)

Keywords: convolutional neural network, handwritten digit recognition, mnist, optimizer, accuracy-efficiency trade-off, lightweight cnn

Introduction

Handwritten digit recognition is one of the earliest and most visible applications of supervised image classification. The task has practical links to bank cheque processing, form digitization, postal code recognition, document analysis, and educational image-processing systems. It has also become a common benchmark for studying machine learning algorithms because the input structure is simple, the target classes are clear, and performance differences can be observed without requiring very large computational resources [1,2].

The role of MNIST (Modified National Institute of Standards and Technology) in this area has changed. Earlier studies used MNIST to demonstrate the strength of feature extraction, neural networks, support vector machines, and convolutional neural networks (CNNs). Later CNN studies reported high MNIST accuracy using architectural variation, augmentation, and training-parameter adjustment [3,4]. However, surveys of MNIST and EMNIST show that MNIST is now a mature and

highly saturated benchmark, with many methods reaching very low error rates [5]. This means that MNIST should not be used alone to claim broad computer-vision novelty, but it can still support controlled baseline analysis, lightweight model comparison, teaching, reproducibility checks, and studies of training-configuration behavior under limited computational cost.

For lightweight CNNs, accuracy alone is not a complete evaluation criterion. A configuration may produce high accuracy but require much longer training time, depend strongly on augmentation, or become unstable when the number of epochs changes. Optimizers such as stochastic gradient descent (SGD) with momentum, Adagrad, Adadelta, Adam, Adamax, Nadam, RMSprop, and Ftrl represent different approaches to gradient-based learning, adaptive learning rates, and momentum behavior [6-8]. These journal sources define the broader optimizer families evaluated in this study and show that optimizer selection remains an active factor in deep learning training. The present comparison is therefore limited to the optimizer families included in the experimental design.

Benchmarking work in deep learning increasingly evaluates training time, convergence, resource behavior, and energy together with accuracy. Framework and architecture benchmarking studies show that accuracy should be interpreted alongside execution time, memory use, model complexity, and computational cost [9-11]. Green AI literature similarly argues that efficiency should be considered as an evaluation criterion alongside predictive performance [12]. These studies reinforce the methodological point that empirical deep-learning evaluation should report performance as a set of trade-offs rather than as a single accuracy number.

Modern empirical machine learning practice also places stronger emphasis on reproducibility and transparent reporting. Reproducibility guidance emphasizes code, data, and checklist practices for reliable machine learning research [13]. Statistical comparison studies also show that benchmark results should be interpreted carefully when repeated trials are limited [14]. Dataset documentation frameworks further encourage authors to state dataset purpose, composition, collection process, evaluation use, and limitations [15].

This study therefore investigates handwritten digit recognition through an accuracy-efficiency question rather than a novelty claim based on raw MNIST accuracy. The central problem is: when lightweight CNNs are used for handwritten digit recognition, how do optimizer choice, epoch count, augmentation, and convolutional parameter settings affect the balance between classification accuracy and execution time? The contributions of this paper are as follows: (a) It provides a structured empirical characterization of lightweight CNN behavior across optimizer type, epoch count, augmentation condition, and convolutional parameter variation. (b) It analyzes the accuracy-efficiency trade-off rather than relying only on peak classification accuracy. (c) It identifies configuration patterns that are stable, efficient, or unsuitable in the evaluated handwritten digit recognition experiments. (d) It discusses the limitations of MNIST-based evaluation and positions the results as practical configuration guidance for lightweight CNN baselines.

The contribution is analytical: it shows how conclusions change when accuracy, execution time, augmentation effect, and training stability are considered together. The article also follows a transparent reporting direction by making the scope of the experiment explicit and by positioning the findings within the evaluated MNIST setting.

Materials And Methods

Dataset and task

The study focuses on handwritten digit recognition using MNIST-style grayscale digit images. The experimental setup used 28 × 28 input images and a split of 42,000 training images and 28,000 testing images. Each image was normalized to the range from 0 to 1. The target labels represent 10-digit classes, from 0 to 9. A sample from the dataset is shown in Figure *1*.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

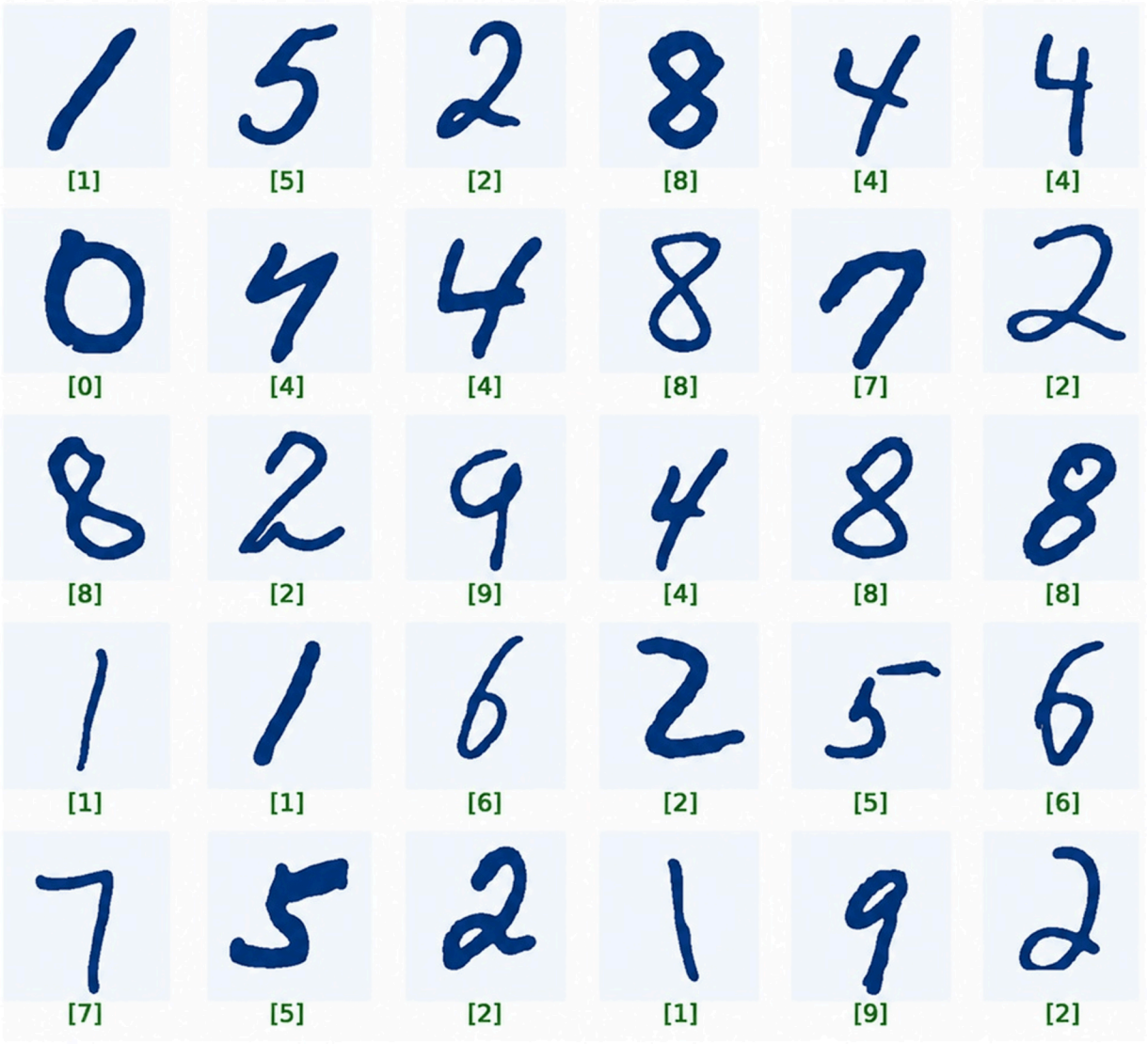


FIGURE 1: Sample handwritten digit images from the MNIST dataset

Image taken from https://www.kaggle.com/tarunkr/digit-recognition-tutorial-cnn-99-67-accuracy.
MNIST, Modified National Institute of Standards and Technology

The workflow follows a standard supervised classification process: data preparation, normalization, optional augmentation, CNN training, prediction, and evaluation using accuracy and execution time. The workflow is summarized in Figure *2*.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

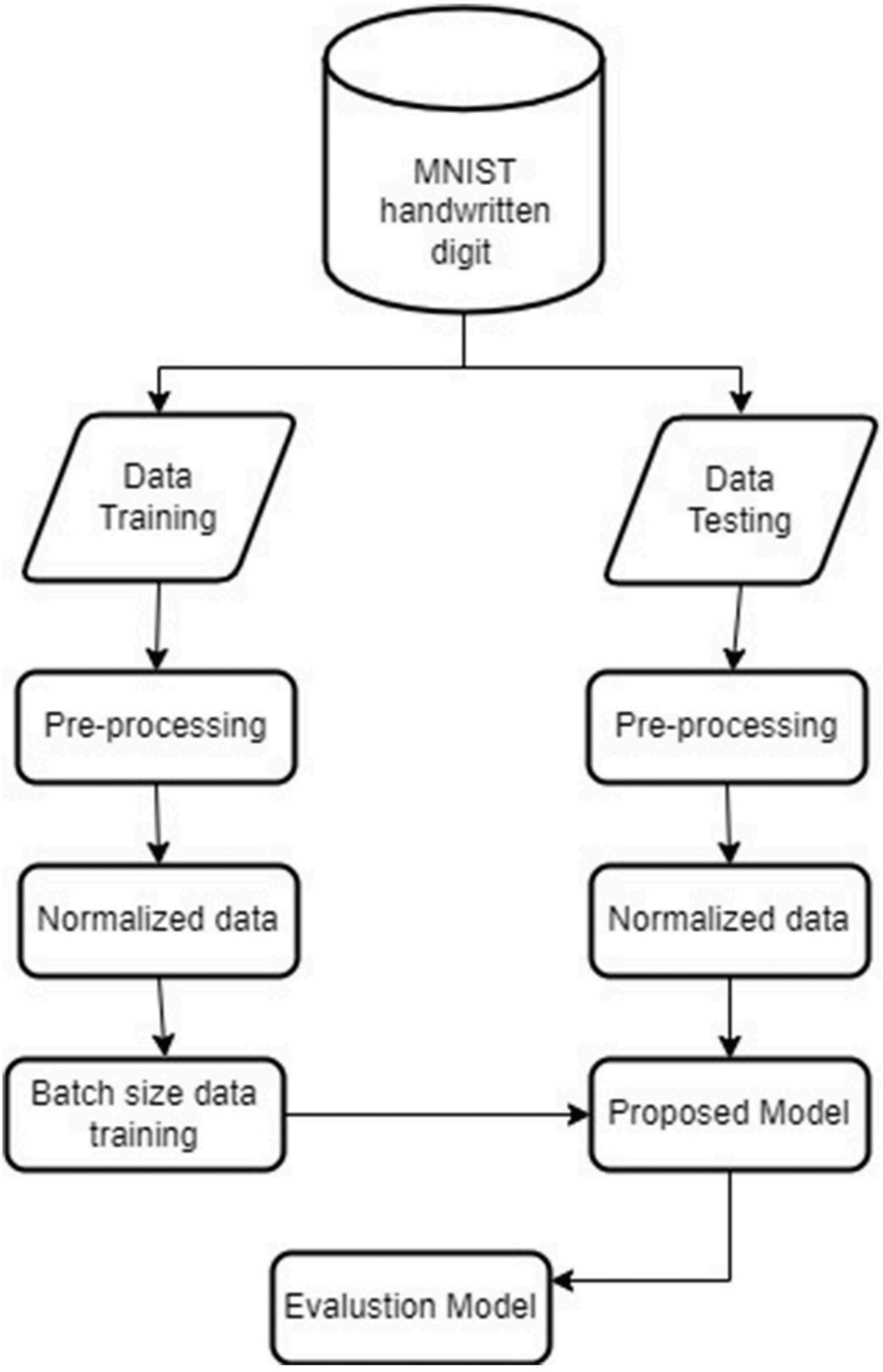

MNIST handwritten digit
Data Training
Data Testing
Pre-processing
Pre-processing
Normalized data
Normalized data
Batch size data training
Proposed Model
Evalustion Model

FIGURE 2: Experimental workflow for the handwritten digit recognition study

CNN configurations

Two five-layer CNN configurations were analyzed. The first model, denoted CNN-5L, used parameter variation across five cases. The second model, denoted CNN-5LS, used a standard parameter setting without the five case variations. For CNN-5L, the varied parameters were kernel size (kz), dilation rate (dr), and stride size (sz). The parameter cases are shown in Table *1*. The documented network design used $28 \times 28 \times 1$ grayscale inputs, five convolutional stages, 32 filters in the convolutional layers, rectified linear unit activation in the convolutional feature extractor, and a softmax output layer for the 10-digit classes. The experimental record documents the kernel-size, dilation-rate, and stride settings reported in Table *1*, together with the optimizer, epoch, augmentation condition, accuracy, execution time, software environment, and hardware information reported below. Parameters that were not documented in the verified experimental record, including batch size, learning-rate overrides, weight-initialization settings, dropout use, dense hidden-layer width, exact augmentation transformation ranges, model parameter count, and floating-point operation count, are therefore not interpreted as controlled factors in this analysis.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

Case	Parameter	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Case-1	kz	3	3	3	3	3
Case-1	dr	1	1	1	1	1
Case-1	sz	2	2	2	2	2
Case-2	kz	3	3	3	3	3
Case-2	dr	2	2	1	1	1
Case-2	sz	1	1	2	2	2
Case-3	kz	5	3	3	3	3
Case-3	dr	2	2	1	1	1
Case-3	sz	1	1	2	2	2
Case-4	kz	5	3	3	3	3
Case-4	dr	1	1	1	1	1
Case-4	sz	1	2	2	2	2
Case-5	kz	5	5	3	3	3
Case-5	dr	2	1	1	1	1
Case-5	sz	1	2	2	2	2

TABLE 1: Convolutional parameter cases used in the CNN-5L configuration

CNN, Convolutional Neural Network

Experimental factors

The experimental design is summarized in Table *2*. It includes model configuration, optimizer or control setting, epoch count, augmentation condition, and performance metrics. The optimizer factor contains eight named optimizers and one baseline condition labeled Control. The control condition is reported separately and is not interpreted as an optimizer family.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

Factor	Levels
Model configuration	CNN-5L with five parameter cases; CNN-5LS with standard parameters
Optimizer/control setting	Adam, Adagrad, SGD, RMSprop, Adadelta, Adamax, Nadam, Ftrl, and control
Epoch count	10, 15, 30, 60, and 120 epochs
Augmentation condition	Before augmentation and after augmentation
Metrics	Classification accuracy in percent and execution time in seconds

TABLE 2: Experimental factors analyzed in the retained matrix

CNN, Convolutional Neural Network; SGD, Stochastic Gradient Descent

The factorial design contains $9 \times 5 \times 5 \times 2 = 450$ paired observations for CNN-5L and $9 \times 5 \times 2 = 90$ paired observations for CNN-5LS. The complete analysis therefore covers 540 observations. Summary figures are used to present the main accuracy, time, optimizer, and augmentation patterns. The augmentation factor is analyzed as a recorded before/after condition. Because the exact augmentation transformations and parameter ranges were not documented in the verified experimental record, the study does not attribute performance changes to individual transformations such as rotation, translation, shear, or zoom.

Analysis strategy

Four analysis questions guide the results:

1. Which configurations achieve the highest accuracy?
2. Which configurations provide high accuracy with lower execution time?
3. How does augmentation affect accuracy across optimizer settings?
4. Which optimizer and parameter settings are stable across epoch values?

The analysis reports observed accuracy and execution time directly. It also uses an accuracy-time trade-off plot and a Pareto frontier. A configuration is on the Pareto frontier if no other configuration has both higher accuracy and lower execution time. Since repeated random-seed trials were not part of the experimental design, the article avoids statistical significance claims and focuses on descriptive empirical characterization.

Results And Discussion

Experimental setup

The results were obtained from an experimental setup using 28×28 grayscale MNIST digit images [1], with 42,000 images for training and 28,000 images for testing. The evaluation was conducted in a Jupyter Notebook environment using TensorFlow and scikit-learn on a Windows 10 machine equipped with an Intel Core i7-10750H processor, 24 GB memory, and an NVIDIA RTX 3050 GPU. Two lightweight CNN configurations were evaluated: CNN-5L with five convolutional parameter cases and CNN-5LS with a standard parameter setting. The training configurations combined

nine optimizer or control settings, five epoch values (10, 15, 30, 60, and 120), and two augmentation conditions. The reported performance metrics are classification accuracy in percent and execution time in seconds, covering 540 observations.

Overall accuracy and execution-time patterns

The highest observed accuracy was 99.50%, achieved by CNN-5LS with Nadam after augmentation, at 120 epochs. This result required 1,011 s. The same standard CNN-5LS configuration with RMSprop reached 99.47% at both 15 and 120 epochs after augmentation, but the execution time was 114 s at 15 epochs and 878.4 s at 120 epochs (Table *3*). This difference is central to the interpretation of the study: peak accuracy alone hides a large computational cost difference.

Model	Setting	Epochs	Accuracy (%)	Time (s)
CNN-5LS	Nadam, after augmentation	120	99.50	1,011.0
CNN-5LS	RMSprop, after augmentation	120	99.47	878.4
CNN-5LS	RMSprop, after augmentation	15	99.47	114.0
CNN-5LS	Adam, after augmentation	30	99.42	222.9
CNN-5LS	RMSprop, after augmentation	30	99.45	229.3
CNN-5LS	Control, after augmentation	30	99.47	269.0
CNN-5L Case-2	Control, after augmentation	10	99.11	76.2
CNN-5L Case-4	Control, after augmentation	15	99.14	121.2

TABLE 3: Representative high-accuracy configurations from the experimental design

CNN, Convolutional Neural Network

Accuracy-efficiency trade-off

Figure *3* plots configurations with at least 90% accuracy and overlays the Pareto frontier. The plot shows that many high-accuracy configurations cluster between 98% and 99.5%, but execution time ranges from tens of seconds to more than 1,000 s. Augmented CNN-5LS settings often reach the highest accuracy, while CNN-5L settings include many faster configurations with moderate to high accuracy.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

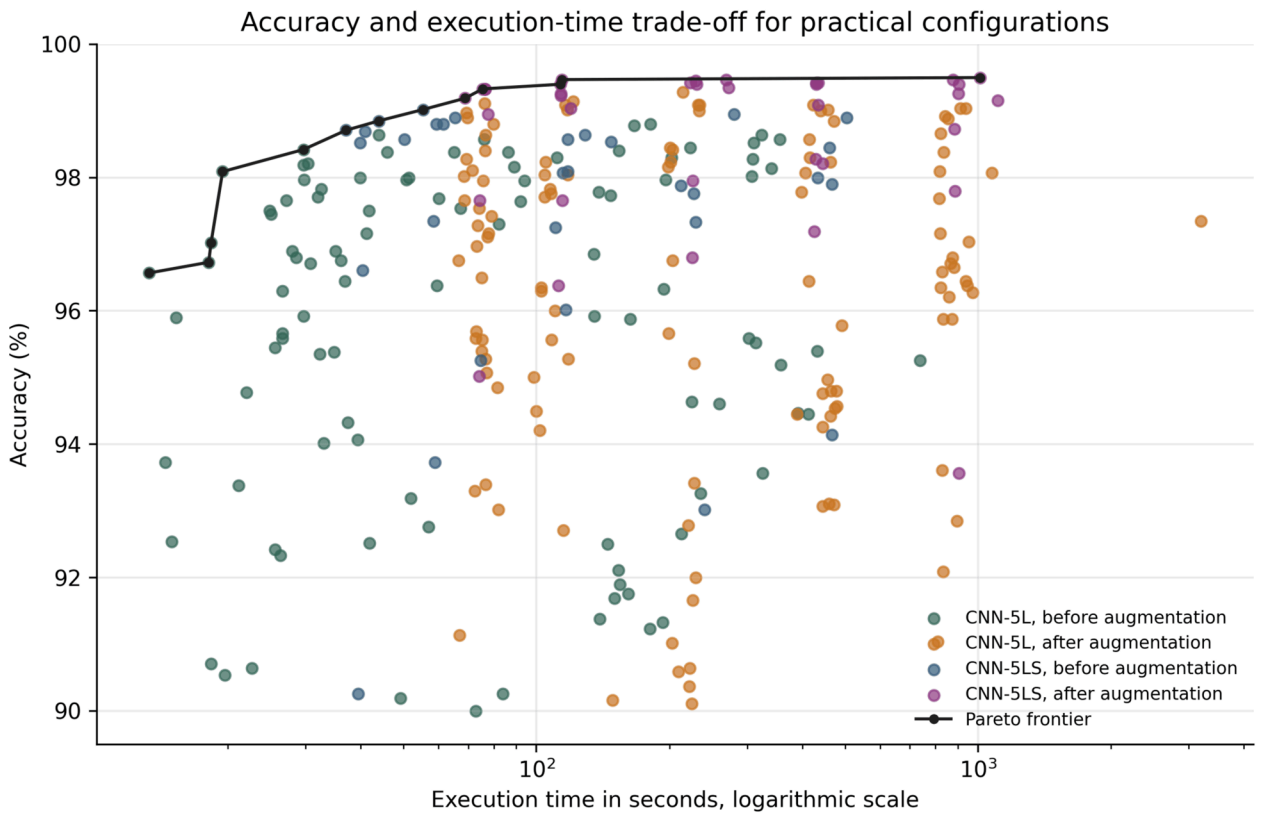


FIGURE 3: Accuracy and execution-time trade-off for practical configurations

CNN, Convolutional Neural Network

The trade-off view changes the interpretation of the experiment. Nadam at 120 epochs produced the highest observed accuracy, but RMSprop at 15 epochs provided nearly the same best-level performance at a much lower execution time. This makes RMSprop a stronger practical choice for lightweight CNN baseline training when computational cost matters.

Optimizer behavior in the standard configuration

Figure *4* shows the after-augmentation behavior of the standard CNN-5LS configuration for the optimizer settings that reached practical accuracy levels. RMSprop was the most consistent named optimizer, remaining between 99.33% and 99.47% across all epoch settings. Adam and Adamax also performed strongly after augmentation, although their before-augmentation results were less stable at higher epochs. Nadam achieved the best single accuracy at 120 epochs but showed a lower value at 60 epochs, which makes it less stable than RMSprop across the full epoch range.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

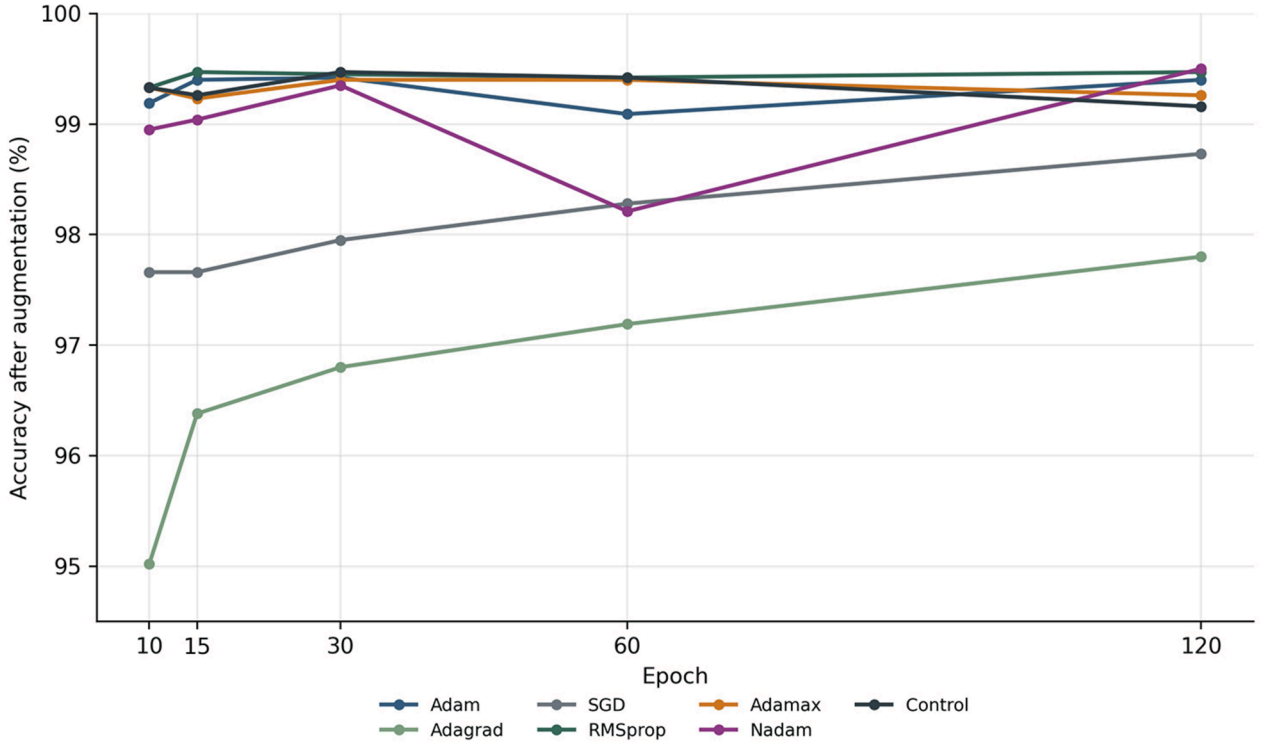


FIGURE 4: Optimizer behavior in the standard CNN-5LS configuration after augmentation

CNN, Convolutional Neural Network; SGD, Stochastic Gradient Descent

Ftrl did not produce meaningful learning in the evaluated configurations, remaining at 11.47% accuracy across CNN-5LS epoch values and across all CNN-5L parameter cases. This value is close to chance-level behavior for 10 classes. Adadelta showed substantial improvement with more epochs in CNN-5LS but did not match the stability of RMSprop, Adam, or Adamax.

Effect of augmentation

Augmentation improved accuracy unevenly across optimizers, as shown in Figure *5*. In the standard CNN-5LS configuration, the mean accuracy gain after augmentation was largest for Adadelta and Nadam, mainly because their before-augmentation values included low or unstable points. Adamax and Adam also gained substantially. RMSprop and SGD had smaller gains because they already performed well before augmentation. Ftrl showed no gain.

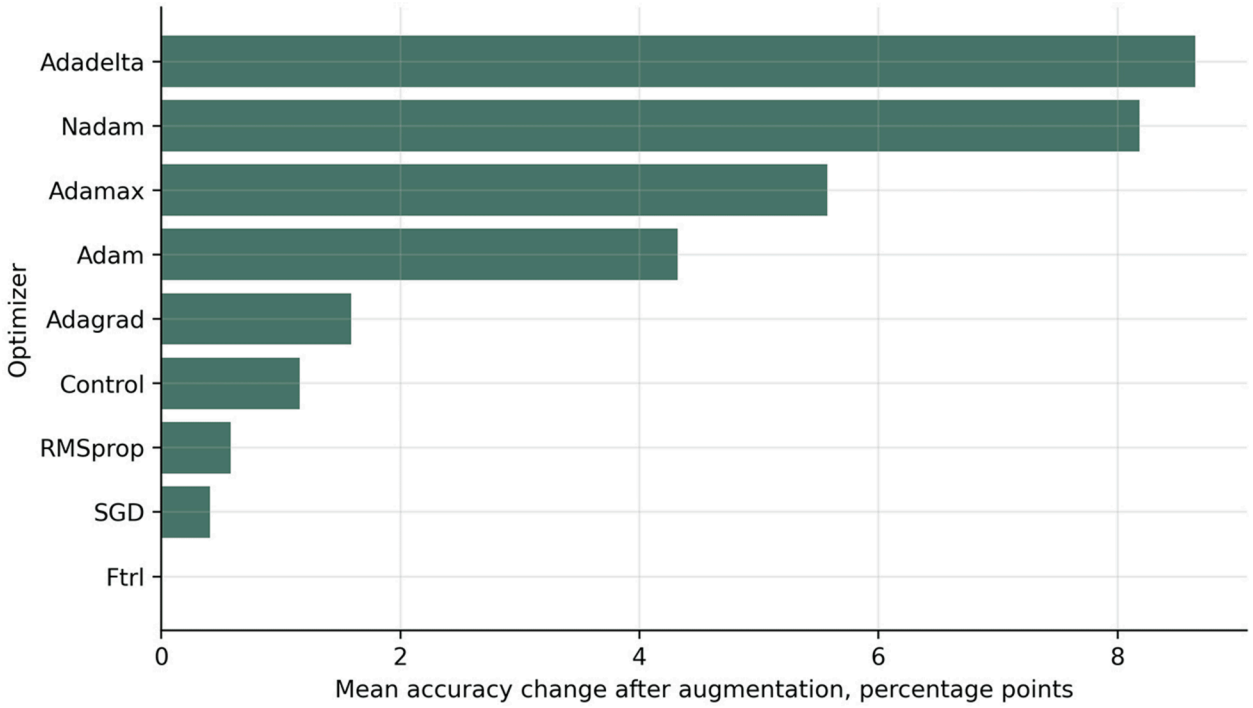


FIGURE 5: Mean augmentation effect in the standard CNN-5LS configuration

CNN, Convolutional Neural Network; SGD, Stochastic Gradient Descent

This result suggests that augmentation should not be interpreted as uniformly beneficial in the same way for every optimizer. It is most useful when the baseline training condition is unstable or underperforming. When the optimizer already produces stable high accuracy, augmentation may still improve robustness, but the marginal accuracy gain can be small compared with the added execution time.

Effect of parameter variation

The CNN-5L results show that convolutional parameter variation did not automatically improve performance. Case-1, which used 3 × 3 kernels, dilation rate 1, and stride 2 across the five layers, was generally the most reliable and efficient case among the named optimizer settings. Cases with larger kernels, higher dilation values, or mixed stride patterns often showed lower or less stable accuracy, especially with Adagrad, SGD, Adadelta, and Ftrl.

This pattern is important because it shifts the conclusion away from "more parameter variation is better." The observed evidence suggests that for the simple MNIST digit task, a compact and regular convolutional setting can be more stable than more aggressive dilation or stride combinations. Parameter variation may reduce execution time or alter receptive-field behavior, but it should be evaluated through the accuracy-time trade-off rather than assumed to improve performance.

Main interpretation

The main finding is that optimizer and configuration effects should be evaluated jointly with execution time. If the goal is the highest observed accuracy, CNN-5LS with Nadam at 120 epochs is the top entry. If the goal is a practical lightweight baseline, CNN-5LS with RMSprop at 15 epochs is more attractive because it reaches 99.47% accuracy in 114 s. These two conclusions are both valid, but they answer different questions.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

This distinction is the core contribution of the study. It characterizes how training choices affect accuracy, time, and stability in a lightweight handwritten digit recognition setting. This framing is more appropriate for a mature benchmark such as MNIST, where peak accuracy alone is no longer a strong novelty claim [5].

Implications for lightweight CNN baselines

Three practical implications follow from the results. First, RMSprop is a strong default candidate for lightweight CNN handwritten digit baselines when stable high accuracy and moderate execution time are required. Second, augmentation should be evaluated with cost in mind. It can rescue weak configurations, but it may add considerable time when the accuracy gain is marginal. Third, convolutional parameter changes such as dilation and stride should not be treated as automatic improvements. In this experiment, the regular Case-1 setting was often more reliable than more varied cases.

The Control condition produced several high-accuracy results. Because this condition is not a named optimizer family, it is analyzed as a baseline condition rather than as part of the optimizer comparison.

Relation to previous studies

Previous MNIST studies often emphasize model accuracy and compare results with earlier approaches [3,4]. Benchmark-oriented studies, however, show that accuracy should be interpreted together with training time, convergence, energy, and resource behavior [2,9-12]. The present study follows the second tradition. It uses MNIST not to claim broad computer-vision novelty, but to analyze lightweight CNN configuration behavior under a controlled experimental matrix.

Modern lightweight architecture families, including MobileNetV2/V3, ShuffleNet, EfficientNet-Lite, and GhostNet, are designed for resource-constrained inference using architectural mechanisms that differ from the simple five-layer CNN baselines evaluated here. The present study therefore should not be interpreted as a direct comparison with those architectures. Instead, it provides configuration-level evidence for compact CNN baselines. A fair comparison with modern lightweight architectures would require new experiments under the same data split, augmentation protocol, hardware, and reporting criteria.

The paper also aligns with reproducibility-oriented reporting. Datasheets and reproducibility checklists encourage clear statements about datasets, training settings, metrics, and limitations [13,15]. This matters because empirical machine learning claims are easy to overstate when the data split, code, hardware, random seeds, or repeated-run behavior is not fully available.

Limitations

This study has several limitations. First, the evaluation is limited to MNIST-style handwritten digit recognition. The results should not be generalized to more complex image-recognition tasks without additional datasets such as EMNIST, Fashion-MNIST, CIFAR-10, or domain-specific OCR data. Second, the experimental design does not include newer optimizers such as AdamW, Lion, MADA-style adaptive frameworks, or memory-efficient Adam variants. The comparison is therefore limited to the optimizer and control settings evaluated in this study. Third, execution time is hardware-dependent. The experiments were conducted on a Windows 10 machine with an Intel Core i7-10750H processor, 24 GB of memory, and an NVIDIA RTX 3050 GPU, so timing comparisons are most meaningful inside the same experimental design.

Fourth, repeated random-seed trials were not included. For this reason, the paper reports descriptive patterns rather than statistical significance. Fifth, implementation-level reproducibility is partial because some settings requested for full replication, including batch size, learning-rate overrides, initialization strategy, exact augmentation ranges, model-size measures, and floating-point operation counts, were not documented in the verified experimental record. These omissions limit the strength of reproducibility and statistical claims. Future work should extend the design with repeated runs, additional datasets, modern lightweight architectures, full software-version reporting, complexity measures, and fully documented executable workflows.

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

Conclusions

This paper presented an accuracy-efficiency characterization of lightweight CNN configurations for handwritten digit recognition. The analysis shows that the best MNIST accuracy and the most practical configuration are not necessarily the same. CNN-5LS with Nadam at 120 epochs achieved the highest observed accuracy of 99.50%, but CNN-5LS with RMSprop at 15 epochs achieved 99.47% with far lower execution time. Augmentation improved several weak or unstable configurations, but its marginal benefit was smaller for already stable optimizers. In the parameter-varied CNN-5L model, Case-1 was generally the most reliable setting, while more aggressive dilation and stride variations did not consistently improve performance.

The study provides practical guidance for lightweight CNN baseline selection: report accuracy and time together, examine optimizer stability across epochs, and avoid treating MNIST peak accuracy as sufficient evidence of methodological novelty. Future work should extend the same accuracy-efficiency analysis to harder handwritten character datasets and release executable training scripts with model-card style documentation.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Muhammad Munsarif, Richmond Ampah-Mensah

Acquisition, analysis, or interpretation of data: Muhammad Munsarif

Drafting of the manuscript: Muhammad Munsarif, Richmond Ampah-Mensah

Critical review of the manuscript for important intellectual content: Muhammad Munsarif, Richmond Ampah-Mensah

Supervision: Muhammad Munsarif

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request.

References

1. LeCun Y, Bottou L, Bengio Y, Haffner P: Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998, 86:2278-2324. 10.1109/5.726791

2. Liu C, Nakashima K, Sako H, Fujisawa H: Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recognition. 2003, 36:2271-2285. 10.1016/s0031-3203(03)00085-2

3. Ali S, Shaukat Z, Azeem M, Sakhawat Z, Mahmood T, Rehman K: An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. Discover Applied Sciences. 2019, 1:1125. 10.1007/s42452-019-1161-5

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w

4. Ahlawat S, Choudhary A, Nayyar A, Singh S, Yoon B: Improved handwritten digit recognition using convolutional neural networks (CNN). Sensors. 2020, 20:3344. 10.3390/s20123344

5. Baldominos A, Saez Y, Isasi P: A survey of handwritten character recognition with MNIST and EMNIST. Applied Sciences. 2019, 9:3169. 10.3390/app9153169

6. Duchi J, Hazan E, Singer Y: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research. 2011, 12:2121-2159.

7. Sun S, Cao Z, Zhu H, Zhao J: A survey of optimization methods from a machine learning perspective. IEEE Transactions on Cybernetics. 2020, 50:3668-3681. 10.1109/tcyb.2019.2950779

8. Wen X, Zhou M: Evolution and role of optimizers in training deep learning models. IEEE/CAA Journal of Automatica Sinica. 2024, 11:2039-2042. 10.1109/jas.2024.124806

9. Elshawi R, Wahab A, Barnawi A, Sakr S: DLBench: A comprehensive experimental evaluation of deep learning frameworks. Cluster Computing. 2021, 24:2017-2038. 10.1007/s10586-021-03240-4

10. Bianco S, Cadene R, Celona L, Napoletano P: Benchmark analysis of representative deep neural network architectures. IEEE Access. 2018, 6:64270-64277. 10.1109/access.2018.2877890

11. Sze V, Chen YH, Yang TJ, Emer JS: Efficient processing of deep neural networks: a tutorial and survey. Proceedings of the IEEE. 2017, 105:2295-2329. 10.1109/JPROC.2017.2761740

12. Schwartz R, Dodge J, Smith NA, Etzioni O: Green AI. Communications of the ACM. 2020, 63:54-63. 10.1145/3381831

13. Pineau J, Vincent-Lamarre P, Sinha K, et al.: Improving reproducibility in machine learning research: A report from the NeurIPS 2019 Reproducibility Program. Journal of Machine Learning Research. 2021, 22:1-20.

14. Dietterich TG: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation. 1998, 10:1895-1923. 10.1162/089976698300017197

15. Gebru T, Morgenstern J, Vecchione B, Vaughan JW, Wallach H, Daumé H III, Crawford K: Datasheets for datasets. Communications of the ACM. 2021, 64:86-92. 10.1145/3458723

How to cite this article:
Munsarif M, Ampah-Mensah R (July 30, 2026) Accuracy–Efficiency Analysis of Optimizer Variants and Convolutional Parameter Configurations for Handwritten Digit Recognition. Cureus J Comput Sci 3 : es44389-026-00193-w. DOI https://doi.org/10.7759/s44389-026-00193-w